In: Proc. of the 2006 IEEE Swarm Intelligence Symposium, pp. 77-84.

HIERARCHICAL CLASSIFICATION OF G-PROTEIN-COUPLED RECEPTORS WITH A PSO/ACO ALGORITHM

Nicholas Holden

Computing Laboratory, University of Kent
Canterbury, CT2 7NF, UK
nh56@kent.ac.uk

Alex A. Freitas

Computing Laboratory, University of Kent
Canterbury, CT2 7NF, UK
A.A.Freitas@kent.ac.uk

ABSTRACT

In our previous work we have proposed a hybrid Particle Swarm Optimisation / Ant Colony Optimisation (PSO/ACO) algorithm for discovering classification rules. In this paper we propose some modifications to the algorithm and apply it to a challenging hierarchical classification problem. This is a bioinformatics problem involving the prediction of G-Protein-Coupled Receptor's (GPCR) hierarchical functional classes. We report the results of an extensive comparison between four versions of swarm intelligence algorithms – two versions based on our proposed algorithm and two versions based on Discrete PSO for discovering classification rules proposed in the literature. The experiments also compared the effectiveness of different kinds of protein signatures when used as predictor attributes, namely Prints, Interpro and Prosite signatures.

I. INTRODUCTION

The large amount of proteomic and genomic data now being produced by modern and efficient sequencing techniques has exceeded the capacity of wet lab experimentation. UniProt is a good example of this. This large protein sequence database contains 205,780 entries from Swiss-Prot which have been manually annotated and verified by curators. It also contains 2,533,011 entries [1] from TrEMBL, which have yet to be manually annotated and so are annotated automatically. It is clear to see the gap between the number of proteins with sequences identified and the number that are closely examined by human curators. The importance of the automatic techniques created to deal with this data is also apparent. Many biologists rely on the data that is automatically annotated and so it is important that the automatic annotation process be accurate, transparent and accountable.

A large sub set of proteins is the G-Protein Coupled Receptor (GPCR) family. GPCR research is an area of intense study due to the success of past drugs that interact with them. The prediction of their function is the topic of this paper. Although there has been work in this field already [2]-[4], the techniques that have been used so far are in general "black box" techniques from the point of view of the user, i.e., they produce predictions that cannot be interpreted by biologists. Examples of these techniques are SVMs (Support Vector Machines) and HMMs (Hidden Markov Models). These techniques tend to produce good classification accuracies, but their output is opaque. To overcome this limitation, in this paper we classify GPCR functions with a hybrid Particle Swarm Optimisation / Ant Colony Optimisation (PSO/ACO) algorithm that discovers IF-THEN classification rules. By comparison with the opaque output produced by SVMs and HMMs, classification rules have the advantage that they tend to be easily interpreted by the user.

Another limitation of the body of work in the area of GPCR function prediction is that the works in this area, in general, do not consider the importance of the hierarchical nature of this data. Indeed, several papers on GPCR prediction address the prediction of only the topmost levels of the class Hierarchy [2]-[4], effectively ignoring that hierarchy. The hierarchy present in GPCR functional data conveys information in its own respect. We believe it is important to try and exploit this property and we show the advantages of doing so in our experiments.

This paper has the following contributions. First, to the best of our knowledge, it is the first paper to apply a swarm intelligence technique to the problem of predicting GPCR function. Second, it proposes important modifications to the original hybrid PSO/ACO classification algorithm, recently proposed and applied to hierarchical classification in our previous work [5]. We describe a drawback of the rule quality measure used in the original algorithm (and also used in several other bio-inspired algorithms), in the context of rules predicting the minority class, and then propose a method of rectifying that rule quality measure in this context. Thirdly it also describes a minimum pheromone limit similar to the MiniMax [6] system used in ACO to improve search.

II. PROTEINS AND G-PROTEIN-COUPLED RECEPTORS (GPCR)

Proteins are the main building blocks of the cell, and perform almost all the functions related to cell activity. Their primary structure, the one decoded from DNA, is formed from a sequence of amino acids which are held together by covalent bonds (strong bonds). This chain is built one amino acid at a time by the ribosome. In order to function a protein usually must fold to form a complex 3D structure. Some parts of protein sequences are found throughout evolution and across different species. These conserved regions are usually important and efficient protein functional building blocks.

GPCRs are proteins involved in signalling. They span cell walls so that they influence the chemistry inside the cell by

sensing the chemistry outside the cell. More specifically, when a ligand (a substance that binds to a protein) is received by the GPCR, it causes the G-proteins to swap chemicals, a biological switch that causes other reactions within the cell to either be inhibited or allowed. Discussions about the detailed working of this system are ongoing. This kind of protein is particularly important for medical applications because it is believed that 40%-50% of current drugs target GPCR activity [7]. They are a prime target for "magic bullet" type drugs as they are directly accessible from the outside of the cell and can influence important processes inside the cell.

III. HIERARCHICAL CLASSIFICATION

Data mining consists of a set of concepts and techniques used to find useful patterns within a set of data [9], [10]. In this project the discovered knowledge is represented as classification rules. A rule consists of an antecedent (a set of attribute values) and a consequent (class):

IF <attrib = value> AND ... AND <attrib = value>
THEN <class>

The consequent of the rule is the class predicted by the rule for the records (examples) where the predictor attributes hold. An example rule might be IF <Salary = high> AND <Mortgage = No> THEN <Good Credit>. This kind of knowledge representation has the advantage of being intuitively comprehensible to the user. This is important, because the general goal of data mining is to discover knowledge that is not only accurate, but also comprehensible [9][10].

In this paper the classes are arranged in a tree structure where each node (class) has only one parent – with the exception of the root of the tree, which does not have any parent and does not correspond to any class. Hierarchical class datasets present new challenges when compared to flat class datasets. The main challenge comes from the extra complexity associated with such datasets, which is due to two main factors. Firstly, many (depending on the depth) more classes must be assigned to the examples. Secondly, the prediction of a class becomes increasingly difficult as deeper levels are considered, due to the smaller number of examples per class. Most of the previous work related to hierarchical classification has been conducted in the field of text mining [11]. By contrast, this paper addresses the hierarchical classification task in bioinformatics. In particular, in this work each class corresponds to a GPCR function.

In this paper the approach used to take advantage of the hierarchy is the divide and conquer principle [11]. In order to explain this approach, let us first introduce some notation. Each node of the class hierarchy is described by a series of digits separated by a "." delimiter, where the first digit is the index of the class at the first hierarchical level (children of the root node), the second digit is the index of the class at the second level, and so on. For instance, in a dataset with four hierarchical levels, the series of digits 2.3.1.2 refers to a GPCR function defined by class 2 at the first level of the hierarchy, class 3 at the second level of the hierarchy, class 1 at the third level, and class 2 at the fourth level. If, say, class 1.X.X.X (where X denotes any digit) is predicted at the first level for a given protein and the tree node for that class has only the child nodes 1.1.X.X and 1.2.X.X, only these two nodes should be considered when predicting the second-level class of that protein. That is, there is no need to consider the possibility of assigning that protein to the children of, say, the class node 2.X.X.X. This holds both during training and test set classification.

This approach has the important advantage of reducing the number of classes to be considered for prediction at every hierarchical level – with the exception of the first level, where there is no previous classification at a higher class level (since there is no higher class level). It also leads naturally to the creation of a hierarchical set of rules, where each node of the class hierarchy is associated with its own modular set of rules. Hence, if the user wants to get insight about the classification of proteins at any particular internal node of the hierarchy, they can be shown the set of rules associated with that specific node, i.e., the rules that will discriminate among the classes corresponding to the children nodes of that internal node. This approach does, however, create the following potential problem of misclassification. If an example is misclassified at a higher node then in general it has no chance of being correctly classified at lower nodes. Techniques have been suggested [12] [13] to correct this specific problem.

A. A Hierarchical Classification Accuracy Measure

In flat classification, the most common method of evaluating the predictive accuracy of a classification-rule discovery algorithm is to generate a rule set from a training set and then apply the rules to a test set comprised of examples with an unknown class. Predictive accuracy can then be calculated as the percentage of test examples which had their class correctly predicted by the rule set. The same process could be carried out for hierarchical classification, but to gain any insight into the performance of the algorithm we have to use a measure of predictive accuracy that is more oriented towards the hierarchical nature of the class hierarchy. One possibility is to report the predictive accuracy at every level of classification. This generates as many accuracy measures as there are classification levels.

However, measuring hierarchical-classification performance purely in terms of accuracy can be misleading. All classifications are not equal in hierarchical classification. In general, classifications at lower (deeper) levels of the class hierarchy are more difficult than classifications at higher (shallower) levels of the class hierarchy. This is because in general the number of training examples belonging to a class is smaller for classes at lower levels than for classes at higher

levels. Hence, a misclassification at a lower level – say predicting class 1.1.1.1 for a protein that actually has class 1.1.1.2 – tends to be more forgivable than a misclassification at a higher level – say predicting class 1.1.1.1 for a protein that actually has class 4.2.3.2. The former is more forgivable for a twofold reason. First, it is more difficult to avoid, due to the greater difficulty of predicting lower level nodes. Second, it misleads the user less than a misclassification at a higher level.

In order to take this into account, we propose to measure the misclassification of hierarchical GPCR functions as follows. First of all, we associate with each edge of the class tree a weight w. The value of this weight is inversely proportional to the number of the hierarchical level. That is, an edge at the first level – i.e., an edge connecting the root node to one of its (first-level) children – has a larger weight than an edge at the second level – i.e., an edge connecting a first-level node to one of its (second-level) children; an edge at the second level has a larger weight than an edge at the third level, and so on. Once each edge in the class tree has been assigned a weight, the "degree of misclassification" of a given protein is computed as follows. Let C_{pred} be the class predicted for a protein and C_{true} be the true class of that protein. Let M be the misclassification degree associated with predicting class C_{pred} for a given protein when the correct class of that protein is C_{true}. M is computed as the summation of the weight of all edges in the path from the class node C_{pred} to the class node C_{true} in the class tree. Note that if $C_{pred} = C_{true}$ then M = 0, since the previously-defined path is an empty set of edges. On the other hand, if $C_{pred} \neq C_{true}$ then M > 0, assuming all edge weights are positive, which is the case in this paper. The meaning of M is illustrated in Figure 1. To keep the figure simple we consider just a two-level hierarchical classification problem.

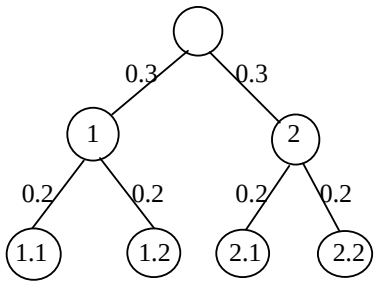

Figure 1: Example of the edge weights used for computing degree of misclassification

In Figure 1, the numbers beside each edge denote the weight of that edge. Suppose C_{pred} = class 1.1 and C_{true}= class 2.1, characterizing a completely wrong prediction. Then the misclassification degree is 0.2 + 0.3 + 0.3 + 0.2 = 1.0. Now suppose C_{pred} = class 1.1 and C_{true}= class 1.2. Then the misclassification degree is 0.2 + 0.2 = 0.4. This is consistent with the fact that this misclassification is much less serious than the previous misclassification example. After all, when C_{pred} = class 1.1 and C_{true}= class 1.2 at least the prediction was correct at the first level (class 1).

In the GPCR dataset addressed in this paper, there are four levels in the class hierarchy. To keep the value of the misclassification degree for a given protein normalized in the range 0 to 1, and to make the edge weight values decrease roughly exponentially as we go down the class tree, we assign the weights 0.26, 0.13, 0.07 and 0.04 to the edges at the first, second, third and fourth level of the class tree, respectively. Hence, a completely wrong classification – such as C_{pred} = class 1.1.1.1 and C_{true}= 3.1.1.1 – has a misclassification degree of 1, as desired.

This measure of misclassification cost is essentially a kind of weighted shortest path distance, a class-distance measure for hierarchical classification discussed in [14]. In addiction, an extensive discussion about the evaluation of hierarchical classification algorithms can be found in [15].

IV. AN OVERVIEW OF THE PSO/ACO ALGORITHM

Here we provide just an overview of the hybrid Particle Swarm Optimization / Ant Colony Optimization (PSO/ACO) algorithm, originally proposed in [5]. For details about this algorithm, readers are referred to that reference. This algorithm was designed to be the first PSO-based classification algorithm to natively support nominal data – i.e., to cope with nominal data directly, without converting a nominal attribute into a numerical one and then apply a mathematical operator to the numerical value. The motivation to natively support nominal data is that by converting a nominal attribute such as *gender* into a numerical attribute (say, mapping *male* into 0 and *female* into 1) we would be introducing an artificial order among the numerical values (1 > 0). Such an order clearly makes no sense in the context of the original nominal values, and mathematical operations applied to this artificial order may generate counter-intuitive results.

The PSO/ACO algorithm achieves a native support of nominal data by combining ideas from Ant Colony Optimisation (Ant-Miner classification algorithm [16]) and Particle Swarm Optimisation [17][18] to create a classification meta heuristic that supports both nominal (including binary as a special case) and continuous attributes.

The hybrid PSO/ACO algorithm discovers a set of rules for each internal node of the class hierarchy. At each internal class node, the algorithm discovers a set of rules of the form IF (conditions) THEN ($class_i$), where $class_i$ is one of the child classes of that internal node. Each rule predicts just a single class, but the set of rules as a whole will predict all classes, because the algorithm guarantees that one or more rules will be discovered predicting each of the child classes.

In order to cope directly with nominal attributes, the hybrid PSO/ACO uses the following approach. A particle contains a

number of pheromone matrices equal to number of categorical attributes in the data set. Each pheromone matrix contains values for pheromones for each possible value that that attribute can take [16] plus a flag value (the indifference flag) indicating whether or not the corresponding attribute is selected to occur in the decoded rule. The particle representation for categorical attributes is shown in graphical form in Figure 2, where each attribute value and the indifference flag are represented as slots in a roulette wheel. This analogy is appropriate for explaining the process of moving the particles with respect to nominal attributes, as discussed next.

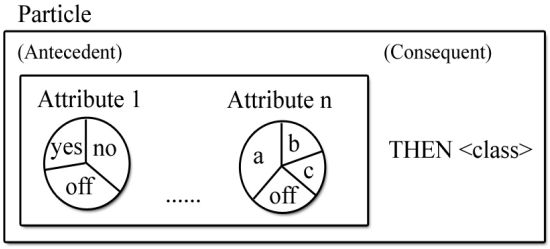

Figure 2: Particle representation for nominal attributes

At each iteration, each categorical attribute in the rule antecedent represented by each particle has its value chosen, in order to give a particle a fixed position and so quality. This is the decoding process. An attribute value is chosen with probability proportional to its pheromone value. This fixed position and so quality is used to update the particle's pheromone matrices in the next iteration. If the new position has a higher quality than any position the particle has ever occupied then it is set as the particle's past best position. To update the values in the pheromone matrices of the current particle, the past best, current and its best neighbour's positions are used. The quality of these three positions, multiplied by individual random learning factors as usual in PSO, are added to the values in the appropriate entries in the pheromone matrices of the current particle. Hence, the mechanism of increasing the pheromone of a given attribute value in the hybrid PSO/ACO corresponds to the mechanism of moving a particle towards that attribute value in conventional PSO. For more details about this process the reader is referred to [5].

After a set of rules has been discovered for each class node, the examples in the test set are classified by using a top-down approach, as follows. The example is shown to the set of rules at the root node, and the system identifies all rules covering that example, i.e. all rules whose antecedent conditions (the IF part of the rule) is satisfied by the attribute values in the example. Out of all rules covering the example, the system chooses the highest-quality one – according to a measure of rule quality to be described later – and assigns the first-level class predicted by that rule to the example. If there are no rules at the root node covering the example, the first-level class assigned to the example is simply the most frequent first-level class in the training set. Let C_1 denote the first-level class assigned to the test example. Next, the example will be classified by the rule set at the node corresponding to class C_1 in the class hierarchy. Similarly, the example will be assigned the second-level class predicted by the highest-quality rule among all rules covering the example, or, if no rule covers the example, the most-frequent second-level class in the training set. And so on, until the example is assigned the leaf-level class.

A. The Modified Rule Quality Measure

The previous version of the PSO/ACO algorithm used the following rule-quality measure to evaluate the predictive accuracy of a candidate rule:

Rule Quality = Sensitivity × Specificity, where

Sensitivity = TP / (TP + FN), Specificity = TN / (TN + FP):

- TP (True Positives) is the number of training cases that have the positive class and satisfy the antecedent of the current rule predicting the positive class
- FP (False Positives) is the number of training cases that have the negative class but satisfy the antecedent of the current rule predicting the positive class
- FN (False Negatives) is the number of training cases that have the positive class but do not satisfy the antecedent of the rule predicting the positive class;
- TN (True Negatives) is the number of training cases that have the negative class and do not satisfy the antecedent of the current rule predicting the positive class.

Note that the sum (TP + TN) represents the total number of correct classifications in the training set, whilst the sum (FP + FN) represents the total number of misclassifications in the training set. This rule quality measure is also used in several other bio-inspired meta-heuristic based classification algorithms – e.g., [16][18][19]. In other words sensitivity is a measure of how well a rule's antecedent covers the examples in the class predicted by its consequent. Specificity is a measure of how well a rule's antecedent avoids covering examples in classes that are not the predicted by the rules consequent. When the proportion of negative and positive examples are well balanced, this rule quality measure works well. However, this rule quality measure does not represent the desirability of a rule as well when the vast majority of the examples belong to the negative class, as explained next.

TABLE 1: EXAMPLE CONFUSION MATRIX (RULE 1)

		Actual Class	
		Positive	**Negative**
Predicted Class	Positive	TP = 5	FP = 0
	Negative	FN = 5	TN = 90

Rule 1's Quality = Sens × Spec = 0.5 × 1 = 0.5

TABLE 2: EXAMPLE CONFUSION MATRIX (RULE 2)

		Actual Class	
		Positive	**Negative**
Predicted Class	Positive	TP = 9	FP = 36
	Negative	FN = 1	TN = 54

Rule 2's Quality = Sens × Spec = 0.9 × 0.6 = 0.54

Consider two example rules with the confusion matrices shown in Tables 1 and 2, where 90 of the examples belong to the negative class and just 10 examples belong to the positive class. Note that rule 2 (Table 2) has a higher Sensitivity x Specificity value than rule 1. However, it is interesting to compute the precision of each of those rules, where precision is usually defined as:

$$\text{Precision} = TP / (TP + FP)$$

Precision is important because it is a direct measure of the confidence or reliability of the rule, when it is applied. Note that, from the point of view of the application of an individual rule, maximizing the number of true negatives (which is incorporated in the use of Specificity as a term of the rule quality) is not so crucial. After all, the rule will be used just to classify examples that are satisfying its antecedent. It is true that Specificity involves not only maximizing the number of true negatives, but also minimizing the number of false positives. However, when the vast majority of examples have the negative class, it is relatively easy to maximize specificity, even though there might be several examples in the category of false positives. In this case, the measure of precision introduces a stronger pressure towards minimizing the number of false positives, because the number of true positives will tend to be relatively low – given that the positive class is the minority class.

Let us now compute the precision of the rules in Table 1 and Table 2. The precision of rule 1 is 5 / 5 = 100%, whilst the precision of rule 2 is just 9 / 36 = 20%. Of course, rule 2 has the advantage of covering a significant higher proportion of examples of the positive class (i.e., significantly higher sensitivity). However, in the above example the price paid for this high sensitivity is too high, the confidence of the rule is simply too low. Hence, intuitively rule 1 seems the best rule, a fact that is not captured by the definition of the Sensitivity × Specificity formula.

Therefore, we propose replacing specificity with precision, when a rule predicts the minority class (as in this example). The hybrid PSO/ACO algorithm uses the following modified quality measure as a particle's fitness when it predicts the minority class.

$$\text{Rule Quality} = \text{Sensitivity} \times \text{Precision}$$

When the majority class is to be predicted the normal rule quality measure is used. So, in the above examples, the new qualities of rules 1 and 2 are $0.5 \times 1 = 0.5$ and $0.9 \times 0.2 = 0.18$, so that rule 1 is considered better than rule 2. We have verified that this modified rule quality measure produces significantly higher test set accuracies during preliminary experiments.

B. Minimum Pheromone Limit

To increase the exploration of the PSO/ACO algorithm, it has been found useful to add a mechanism to allow exploration even after the population has converged. This is achieved by limiting the minimum value possible for an attribute-value's pheromone entry. This means that there is never a probability of 0 of choosing an attribute-value. This is conceptually similar to the ACO MiniMax system [6] as due to normalisation there is a maximum pheromone limit of 1 and also a minimum limit defined by the minimum pheromone limit. It is also conceptually similar to setting a maximum velocity value in conventional Binary PSO [20], which has the effect of introducing a minimum value for the probability of choosing a certain binary value. The optimal value of this lower bound – like most parameters in PSO and other bio-inspired algorithms – tends to be problem dependent. In this work we used the lower bound of 0.1% for each pheromone value. This threshold value was empirically determined in our preliminary experiments, but we make no claim that this is an optimal value. Parameter optimisation is a topic left for future research.

V COMPUTATIONAL EXPERIMENTS

A. Creation of the Data Sets Used in the Experiments

The classes to be predicted in the data sets used in our experiments are the functional classes of GPCRs. These functional classes are given unique hierarchical indexes by [8]. Records have up to 5 class levels, but only 4 levels are used in the datasets, as the data in the 5th level is too sparse for training – i.e., in general there are too few examples of each class at the 5th level. In any case, it should be noted that predicting all the first four levels of GPCR's classes is already a challenging task. Indeed, most works on the classification of GPCRs limit the predictions to just the topmost or the two topmost class levels (families and subfamilies but not groups etc) [2]-[4].

The data used in our experiments was derived from UniProt [21] and GPCRDB [8]. UniProt contains sequence data with a very rich annotation. It also has cross references for other major biological databases such as Prosite, Prints, Interpro (see below). It was extensively used in this work as a source of data for creating the data sets used in our experiments. Only the UniProtKB/Swiss-Prot was used as a data source, as it contains a higher quality, manually annotated set of proteins.

We did experiments with three different kinds of predictor attributes, each of them representing a kind of "protein signature". The three kinds of predictor attributes used in our experiments are: FingerPrints from the Prints [22] database, Prosite [23] patterns and Interpro [24] entries. In essence, the protein signatures associated with these databases have the following characteristics. Prosite patterns are regular expressions describing short fragments of protein sequences.

Such patterns are especially good at detecting things like catalytic sites in enzymes. The regular expressions employed are good at detecting such highly conserved functional regions, as they do not allow partial hits. However due to this rigidity there tend to be a large number of false negatives [25]. Prints contains a set of motifs in each entry, along with descriptions. FingerPrint signatures are different from Prosite entries in that they use multiple sets of amino acid frequency matrices to try and identify an unknown protein rather than just one motif. Note that these motifs are ordered. Another difference is that Prosite's patterns usually correspond to functional regions, whilst it is often the case that a Prints motif refers only to a highly conserved region with no specific function. Interpro integrates several protein identification databases into one.

For each of these three kinds of protein signatures we created a separate dataset, using just that kind of protein signature as the predictor attributes. This allows to compare the results of an algorithm across the three different kinds of protein signatures, to determine whether one of them constitutes a better kind of predictor attribute (in the sense of maximizing the predictive accuracy of an algorithm) than the others. In each of the three datasets, each protein signature was encoded as a binary attribute, where 1 indicates the presence of a protein signature and 0 the absence.

After the initial creation of the data sets, each data set had many duplicate examples (proteins). This happens because often a protein has several variations (produced by mutations in the coding DNA) and each of these variations is stored as a separate entry in Uniprot. Once these proteins are represented by sets of motifs (Prosite, FingerPrints or Interpro signatures), this high level representation tends to lose the details of the variations of a given protein. It is in fact the purpose of these signatures to generalise across protein families. Hence, proteins which are somewhat different from each other in terms of sequence can be represented by the same set of attribute-values, creating many duplicate records. All duplicate examples were removed to avoid the unfair situation where the same example might be included in both the training set and the test set.

After considering the duplication problem, the size of each data set is as follows:

- FingerPrints: 5577 and 338 examples before and after duplicate removal, 281 attributes, 11, 43, 78 and 83 classes at the 1st, 2nd, 3rd and 4th level.
- Prosite: 6462 and 194 examples before and after duplicate removal, 127 attributes, 11, 37, 42 and 12 classes at the 1st, 2nd, 3rd and 4th level.
- Interpro: 7623 and 584 examples before and after duplicate removal, 448 attributes, 14, 49, 98 and 84 classes at the 1st, 2nd, 3rd and 4th level.

B. Experimental Methodology

In all the experiments reported in the next section, the results were produced by a 10-fold cross-validation procedure [9]. We report results concerning two measures of predictive accuracy. First, Tables 3, 4, and 5 report the standard measure of classification accuracy (followed by its standard deviation) for each level of the GPCR class hierarchy. The standard classification accuracy is simply the number of correctly classified test examples divided by the total number of test examples. Tables 3, 4 and 5 report results for the 3 datasets used in our experiments, using as predictor attributes FingerPrint signatures, Interpro entries and Prosite patterns, respectively. Second, Table 6 reports the results according to the misclassification cost measure based on the distance between predicted and actual class nodes in the class hierarchy, as explained earlier.

The experiments involve a comparison between four different algorithms, as follows. PSO/ACO_Hier is the algorithm described in section 4, producing a rule set for each internal node of the class hierarchy and classifying the test examples in a top-down fashion. PSO/ACO_flat is a flat-classification version of that algorithm, where the basic PSO/ACO algorithm is used to produce a single rule set that directly assigns to a test example a bottom level class. Note that by assigning a 4th level class to an example the system is automatically assigning to that example classes at the 1st, 2nd and 3rd level of the hierarchy as well. DPSO_flat is the Discrete PSO algorithm proposed in [18], which was also used to produce a single rule set that directly assigns to a test example a bottom-level class. Note that, unlike the hybrid PSO/ACO algorithm, DPSO transforms nominal attributes into numerical, discrete attributes, and so introduces an artificial ordering among the originally nominal values. DPSO_Heir is a new version of the algorithm proposed in [18] which performs hierarchical classification in the same sense as PSO/ACO_Hier, i.e., producing a rule set for each internal node of the class hierarchy and classifying the test examples in a top-down fashion. For both versions of the DPSO algorithm the values of χ=0.73 (constriction coefficient), $\varphi_1 = \varphi_2 = 2.05$ (social and personal learning factors) were used as is standard in the literature [26]. We compared DPSO and PSO/ACO using the same seeding method and using the same modified rule quality measure.

C. Computational Results

TABLE 3: PREDICTIVE ACCURACY (%) WITH PRINTS ATTRIBUTES

	PSO/ ACO_Flat	**DPSO_Flat**	**PSO/ ACO_Hier**	**DPSO_Hier**
Level: 1	75.65±6.82	75.65±6.82	89.64±6.75	88.62±8.37
Level: 2	34.99±9.09	33.09±8.12	63.44±6.42	62.61±6.28
Level: 3	24.46±5.18	23.22±4.59	45.18±6.61	43.79±9.12
Level: 4	24.23±9.6	24.79±6.28	33.76±8.79	32.09±10.28

TABLE 4: PREDICTIVE ACCURACY (%) WITH INTERPRO ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	55.86±4.64	55.86±4.64	86.31±3.18	86.99±4.27
Level: 2	27.47±4.81	26.62±5.14	72.33±2.84	70.58±4.2
Level: 3	20.74±3.38	20.97±3.04	48.02±7.02	47.53±6.98
Level: 4	30.73±7.55	31.31±7.96	36.52±11.3	35.81±7.61

TABLE 5: PREDICTIVE ACCURACY (%) WITH PROSITE ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	32.27±9.92	32.27±9.92	73.03±8.51	73.54±8.6
Level: 2	6.39±5.96	6.39±5.96	37.49±11.47	37.66±13.3
Level: 3	3.73±4.82	3.98±5.34	22.13±15.58	22.02±15.5
Level: 4	NA	NA	NA	NA

TABLE 6: MISSCLASSIFICATION COST BASED ON DISTANCES IN CLASS TREE

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
GPCR Prints	45.95±5.42	46.8±5.17	25.06±5.66	26.92±7.15
GPCR InterPro	59.87±3.61	59.16±3.55	22.45±3.38	23.31±4.94
GPCR Prosite	78.13±7.45	78.06±7.43	38.56±6.68	38.67±7.15

TABLE 7: NUMBER OF RULES DISCOVERED WITH PRINTS ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	6.9±0.32	6.9±0.32	7.0±0.47	6.9±0.32
Level: 2	26.8±1.99	26.7±1.95	24.8±1.48	24.6±2.27
Level: 3	36.2±3.08	39.6±2.99	28.8±1.32	29.2±1.03
Level: 4	22.6±2.37	22.0±1.83	15.1±1.37	15.9±1.37

TABLE 8: NUMBER OF RULES DISCOVERED WITH INTERPRO ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	12.1±1.45	10.4±0.84	11.4±1.51	10.4±0.7
Level: 2	41.0±2.11	41.1±1.52	36.2±1.4	36.9±0.99
Level: 3	59.3±2.67	58.6±2.8	50.0±2.0	50.3±1.64
Level: 4	18.7±2.0	18.5±1.9	15.8±1.4	17.7±1.57

TABLE 9: NUMBER OF RULES DISCOVERED WITH PROSITE ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	7.4±0.7	7.4±0.7	7.4±0.7	7.4±0.7
Level: 2	18.5±1.35	18.2±1.69	16.0±1.15	16.9±1.29
Level: 3	11.7±1.16	12.0±1.15	9.7±0.82	10.8±0.79
Level: 4	NA	NA	NA	NA

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	6.28±0.9	8.03±1.24	6.43±0.63	7.35±0.57
Level: 2	6.93±0.98	7.94±0.92	5.89±0.42	6.69±1.22
Level: 3	4.23±1.98	5.75±2.02	4.14±0.67	5.07±0.67
Level: 4	3.07±2.1	5.43±1.65	2.39±0.22	2.32±0.21

TABLE 11: NUMBER OF TERMS PER RULE WITH INTERPRO ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	4.09±0.4	5.06±0.51	4.36±0.67	4.98±0.47
Level: 2	5.81±0.91	7.89±0.53	5.03±0.35	5.14±0.49
Level: 3	1.88±1.0	3.96±1.73	3.56±0.54	4.02±0.51
Level: 4	2.65±1.35	5.09±1.14	2.64±0.23	2.46±0.21

TABLE 12: NUMBER OF TERMS PER RULE WITH PROSITE ATTRIBUTES

	PSO/ ACO_Flat	DPSO_Flat	PSO/ ACO_Hier	DPSO_Hier
Level: 1	5.03±0.86	6.76±0.96	5.01±0.81	6.69±0.94
Level: 2	6.07±1.27	7.96±0.69	5.76±0.63	5.81±0.48
Level: 3	4.43±2.11	5.07±1.65	3.73±0.43	3.45±0.35
Level: 4	NA	NA	NA	NA

It is clear to see, from the results in Tables 3 through 6, that the divide and conquer approach (associated with algorithms PSO/ACO_Hier and DPSO_Hier) obtains a considerably higher predictive accuracy than simply predicting bottom-level classes. This is due to the way in which the problem is split into manageable chunks by the divide and conquer approach whilst taken as a whole it is a difficult, many class problem, sometimes with very few positive examples (as discussed earlier). Overall, there is no significant difference in the predictive accuracy of PSO/ACO_Hier and DPSO_Hier. The cells with NA in them refer to the fact that there were not enough examples present to train the classification system on.

Still concerning predictive accuracy, it is also clear that, as predictor attributes for GPCRs classification, Prosite patterns are not as effective as Interpro entries or FingerPrints signatures. Interpro and Prints attributes were competitive with each other. Although it should be noted that they have different strengths and weaknesses. It was to be expected that Interpro entries would produce some of the best results, purely because the Interpro database tends to cover more proteins than either of the other identification systems (as it combines these and other systems). On the other hand, one potential disadvantage of Interpro attributes is that their broader coverage may cause it to loose some accuracy as classifying a wider array of proteins is a harder task. Although Prints does not cover as many proteins as Interpro, the ones it does cover seem to be classified well.

In Table 4 the accuracies actually increase at the bottom level during flat classification. At first glance, this is unexpected as it is impossible for the hybrid PSO/ACO algorithm to correctly classify an example at the bottom level once an incorrect classification has been made at a higher level. However, this is not an error. Some GPCRs have their functional classes specified only up to the third level (or shallower), and not up to the fourth level. If a significant fraction of those GPCRs are misclassified at the third level, the classification accuracy at the third level can actually be smaller than the classification accuracy at the fourth level.

Let us now consider the simplicity of the discovered rule sets – see Tables 7 through 12. In general, the smaller a rule set (the smaller the number of rules and the number of terms per rule), the simpler that rule set is. A simpler rule set facilitates the interpretation of the discovered rules, and therefore is a desirable result [9]. It should be noted that PSO/ACO produces rules with slightly fewer terms in general, by comparison with DPSO. This small effect is observed both in the flat and the hierarchical versions of the two algorithms.

VI. CONCLUSIONS AND FUTURE RESEARCH

The main contributions of this paper are as follows. First, we proposed some modifications to the original PSO/ACO algorithm and justified them. Second, we have clearly shown the advantages of considering the class hierarchy when attempting to classify GPCRs. These results were obtained in an extensive comparison with two versions of a flat classification algorithm and two versions of a hierarchical classification algorithm. Third, the results reported have also shown that Interpro and Prints attributes are much more effective for predicting GPCR ligand binding classes than Prosite attributes.

Future research will involve experiments with other kinds of biological data and with mixed nominal/continuous attributes. Another research direction is to develop a PSO/ACO classification algorithm that is truly hierarchical. Note that although the current algorithm produces a hierarchical rule set, it only considers the flat classification problem during the run of the algorithm at each node in the class tree. It may be useful to try and take advantage of the rule set previously discovered within the hierarchy and so produce an algorithm that is more "aware" of the class-hierarchy structure.

REFERENCES

[1] UniProt Release Notes, ftp://ftp.ebi.ac.uk/pub/databases/UniProt/relnotes.txt, Visited on Jan. 2006.

[2] P.K. Papasaikas, P.G. Bagos, Z.I. Litou, S.J. Hamodrakas. *A novel method for GPCR recognition and family classification from sequence alone using signatures derived from profile hidden Markov models.* SAR QSAR Environ Res, 14(5-6),pp. 413-20, Oct-Dec, 2003.

[3] M. Bhasin, G.P. Raghava. *GPCRpred: an SVM-based method for prediction of families and subfamilies of G-protein coupled receptors.* Nucleic Acids Res. 1, 32(Web Server issue),pp. 383-9, Jul 2004.

[4] R. Karchin, K. Karplus, D. Haussler. *Classifying G-protein coupled receptors with support vector machines.* Bioinformatics, 18(1):pp. 147-59, 2002 Jan.

[5] N. Holden, A.A. Freitas. *A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data*. Proc. 2005 IEEE Swarm Intelligence Symp., pp. 100-107, 2005.

[6] M. Dorigo, T.S. Stützle, Ant Colony Optimization. MIT Press. 199. 2004

[7] D. Fillmore, *It's a GPCR world, Modern drug discovery*, vol. 11, issue 7, pp 24-28, Nov 2004

[8] GPCR, *http://www.gpcr.org/7tm/*, Visited on Jan. 2006.

[9] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools*, 2nd Ed. Morgan Kaufmann Publications, 2005.

[10] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth. *From data mining to knowledge discovery: an overview*, Advances in Knowledge Discovery and Data Mining, AAAI/MIT, pp. 1-34, 1996.

[11] A. Sun, E.-P. Lim, *Hierarchical Text Classification and Evaluation*, Proc. 2001 IEEE ICDM (Int. Conf. on Data Mining), pp. 521-528, 2001

[12] S. T. Dumais, H. Chen. *Hierarchical classification of web content*. In Proc. of the 23rd Int'l ACM Conf. on Research and Development in Information Retrieval (SIGIR). pp. 256—263. August 2000.

[13] A. Sun, E.-P. Lim, W. Keong Ng, J. Srivastava. *Blocking Reduction Strategies in Hierarchical Text Classification*. IEEE Trans. Knowl. Data Eng. 16(10): pp. 1305-1308 .2004.

[14] H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, and J. Struyf, *Hierarchical Multi-Classification*. SIGKDD Workshop on Multi-Relational Data Mining (MRDM-2002), pp. 21—35, 2002.

[15] A. Sun , E.P. Lim. *Hierarchical Text Classification and Evaluation*. Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 521-528. 2001.

[16] R.S. Parpinelli, H.S. Lopes, A.A. Freitas. *Data Mining with an Ant Colony Optimization Algorithm*, IEEE Trans. on Evolutionary Computation, special issue on Ant Colony algorithms, 6(4), pp. 321-332, Aug 2002.

[17] J. Kennedy, R. C. Eberhart, with Y. Shi. *Swarm Intelligence*, San Francisco: Morgan Kaufmann/ Academic Press, 2001.

[18] T. Sousa, A. Silva, A. Neves, *Particle Swarm based Data Mining Algorithms for classification tasks*, Parallel Computing 30, pp. 767–783, 2004.

[19] R.T. Alves, M.R. Delgado, H.S. Lopes, A.A. Freitas. *An artificial immune system for fuzzy-rule induction in data mining.* Proc. Parallel Problem Solving from Nature (PPSN-2004), LNCS 3242, pp. 1011-1020, 2004.

[20] J. Kennedy, R. C. Eberhart. *A discrete binary version of the particle swarm algorithm.* Proceedings of the 1997 Conference on Systems, Man, and Cybernetics, pp. 4104-4109. 1997.

[21] UniProt, *http://www.ebi.UniProt.org/*, Visited on Jan. 2006.

[22] Prints, *http://umber.sbs.man.ac.uk/dbbrowser/PRINTS/*, Visited on Jan. 2006.

[23] ProSite, *http://us.expasy.org/prosite/*, Visited on Jan. 2006.

[24] Interpro, *http://www.ebi.ac.uk/interpro/*, Visited on Jan. 2006.

[25] J. McDowall, *InterPro: Exploring a Powerful Protein Diagnostic Tool*, ECCB05, Tutorial, pp 14, 2005.

[26] M. Clerc, J. Kennedy, *The particle swarm-explosion, stability and convergence in a multidimensional complex space*, IEEE Transactions on Evolutionary Computation 6 (1). 2002.